INFORMATION OF COMPUTATION 82, 323–349 (1989)

Trade-Off among Parameters Affecting Inductive Inference

RŪSIŅŠ FREIVALDS

Computing Centre, Latvian State University, Raiņa bulvaris 29, 226250, Riga, USSR

CARL H. SMITH*

Department of Computer Science and Institute for Advanced Computer Studies, The University of Maryland, College Park, Maryland 20742

AND

MAHENDRAN VELAUTHAPILLAI

Department of Computer Science, Georgetown University, Washington, DC 20057

This paper is concerned with the algorithmic learning, by example in the limit, of programs that compute recursive functions. The particular focus is on the relationship of three parameters that effect inferribility: the number of experimental trials, the plurality of approaches to the particular learning problem, and the accuracy of the final result. Each of these parameters has been examined extensively before. However, the precise characterization of the three-way interaction between these parameters is still not known. This paper makes significant progress toward a complete solution.

I. INTRODUCTION

This paper is concerned with the algorithmic learning, by example in the limit, of programs that compute recursive functions. The particular focus is on the relationship of three parameters that effect inferribility: the number of experimental trials, the plurality of approaches to the particular learning problem, and the accuracy of the final result. Each of these parameters has been examined extensively before. However, the precise characterization of the three-way interaction between these parameters is still not known. This paper makes significant progress toward a complete solution.

* Supported in part by NSF Grant CCR-8701104 and NSA OCREAE Grant MDA904-85-H-0002.

0890-5401/89 $3.00

In what follows, we introduce the necessary terminology and review the relevant literature. For an introduction to the topic of inductive inference, see (Angluin and Smith, 1983). The interest in the field shown by computer scientists is essentially due to artificial intelligence considerations (Angluin and Smith, 1987). We now proceed to develope notation and to discuss the fundamental concepts behind inductive inference.

Members of $\mathbb{N}$, the natural numbers, will serve as program names. The *positive* natural numbers will be denoted by $\mathbb{N}^+$. $\varphi_0, \varphi_1, \ldots$ is an *acceptable programming system* (Machtey and Young, 1978) containing all and only the partial recursive functions of a single argument. The acceptability means that certain natural properties hold for the chosen enumeration of the partial recursive functions. Program i computes the function φ_i. f and g will be used to denote recursive functions for which no program for computing them is yet known. $\subseteq$ denotes *subset* and $\subset$ denotes *proper subset*. The learning will be performed by *inductive inference machines* (abbreviated: IIM) that input the graph of a recursive function and output programs intended to compute the function generating the input. Suppose an IIM M is given the graph of f as input. We may suppose without loss of generality that f is given in its natural order $(f(0), f(1), \ldots)$ to M (Blum and Blum, 1975). M will output a (possibly infinite) sequence of programs $p_0, p_1, \ldots$, each of which may or may not compute f. M is said to converge on input from f (written: $M(f)\downarrow$) iff *either* the sequence $p_0, p_1, \ldots$ is finite and nonempty *or* there is an n such that for all $n' \geqslant n$, $p_{n'} = p_n$. $M(f)\downarrow = p_n$ means that *either* the sequence of output programs is finite with length $n+1$ *or* all but the first n programs in the sequence are precisely p_n. $M(\sigma)$ denotes the most recent output, if any, produced by M on input from the finite sample σ.

Gold (1967) introduced a criterion of successful inference called "identification in the limit." This notion will be called EX identification. An IIM M *EX-infers* f (written $f \in EX(M)$) iff $M(f)\downarrow p$ and $\varphi_p = f$. Each IIM will EX infer some set of recursive functions. EX denotes the class of such sets, e.g., $EX = \{S \mid (\exists M)[S \subseteq EX(M)]\}$. EX stands for "explain," a term consistent with the philosophical motivations for the study of inductive inference, see (Case and Smith, 1983).

For EX inference, the machine must produce a program that is correct on all inputs. This makes the inference more difficult, or perhaps as suggested in (Valiant, 1984), impractical. A partial recursive function ψ is an *n-variant* of a recursive function f (written $\psi = {}^n f$) iff the cardinality of $(\{x \mid \psi(x)\uparrow\} \cup \{x \mid \psi(x)\downarrow \neq f(x)\}) \leqslant n$. ψ is a *finite variant* of f (written $\psi = {}^\star f$) iff $(\{x \mid \psi(x)\uparrow\} \cup \{x \mid \psi(x)\downarrow \neq f(x)\})$ is finite. For any $a \in \mathbb{N} \cup \{\star\}$, an IIM M EX^a-*infers* f (written $f \in EX^a(M)$) iff $M(f)\downarrow p$ and $\varphi_p = {}^a f$. Similarly, for any $a \in \mathbb{N} \cup (\star)$, $EX^a = \{S \mid (\exists M)[S \subseteq EX^a(M)]\}$. Note that $EX^0 = EX$. $EX^\star$ inference was introduced in (Blum and Blum,

1975) and EX^a inference, for $a \neq 0, \bigstar$, was introduced in (Case and Smith, 1983). In the inequalities used to state the hypothesis of some of the theorems below, $\bigstar$ is considered to be greater than any member of $\mathbb{N}$.

Although counting the number of mind changes an IIM makes before converging is not an abstract measure of the complexity of inference (Daley and Smith, 1986), it does provide a reasonable estimate for implemented inference systems. Consequently, the number of mind changes made by inference machines has received considerable attention (Barzdin and Freivalds, 1972; Case and Smith, 1983; Case and Ngo Manguelle; Jantke, 1979; Kinber, 1977; Smith, 1982; Velauthapillai, 1986; Wiehagen *et al.*, 1984). A subscript b on the class name indicates a success criterion where the IIM converges after no more than b changes of conjecture. If $b = \bigstar$ then the IIM is allowed finitely many mind changes. Formally, for $a, b \in \mathbb{N} \cup \{\bigstar\}$, an IIM M EX_b^a-infers f (written $f \in EX_b^a(M)$) iff $M(f)\downarrow p$ in at most b changes of conjecture (mind changes) and $\varphi_p = {}^a f$. For $a, b \in \mathbb{N} \cup (\bigstar)$, $EX_b^a = \{S \mid (\exists M)\, S \subseteq EX_b^a(M)\}$. Consequently, $EX = EX_\bigstar$. A fundamental relationship between anomalies and mind changes is given by: $EX_b^a \subseteq EX_d^c$ iff $[a \leqslant c$ and $b \leqslant d]$ (Case, 1983).

The Blums (1975) constructed two IIMs which inferred classes whose union was not inferrible by any IIM. Subsequently, this result was extended to arbitrarily large finite unions (Smith, 1982). A set of functions S is inferred by the team $M_1, M_2, \ldots, M_n$ if for each $f \in S$ there is an $1 \leqslant i \leqslant n$ such that $f \in EX(M_i)$ (Smith, 1982). This fact will be denoted by $S \subseteq EX(M_1, M_2, \ldots, M_n)$. For $a, b \in \mathbb{N} \cup (\bigstar)$, $C(n, EX_b^a) = \{S \mid (\exists M_1, M_2, \ldots, M_n)\, S \subseteq EX(M_1, M_2, \ldots, M_n)\}$. Different members of the team will infer different member of S. Given an $f \in S$ it is impossible to tell which team member(s) will actually infer f. A team is considered to have output a program when one its members produces an output. In (Smith, 1982) it was shown that $C(n, EX_b^\star) \subseteq C(n', EX_{n'}^\star)$ iff $[n' \geqslant n$ and $n' \cdot (b'+1) \geqslant n \cdot (b+1)]$ and $C(n, EX_\bigstar^a) \subseteq C(n', EX_\bigstar^{a'})$ iff $n' \geqslant n \cdot (1 + \lfloor a/(a'+1) \rfloor)$. These resuts were used, with the results of (Pitt, 1984) relating probabilistic inference precisely with team inference, in (Pitt and Smith, 1988) to reveal the trade-offs between pluralism and plurality.

For implemented inference systems, both the number of mind changes and the number of anomalies tolerated are typically very small. The previously discovered results say nothing about these restricted cases. The goal of this paper is to find an informative and nontrivial predicate P such that $C(n, EX_b^a) \subseteq C(n', EX_{b'}^{a'})$ iff $P(n, n', a, a', b, b')$ holds. The positive partial results that we do obtain depend on simulating one inference team by another. The intent of the simulation is for one machine to infer every function that the team did. The next section gives examples of two basic techniques that are used in all the simulations that follow.

Most of the results presented in the preceding papers consisted of

necessary conditions for two classes to be incomparable. Smith (1982) obtained necessary and sufficient conditions for special cases of the above problem. (Special cases considered were $a = a' = \star$, $m \leqslant n$; and $b = b' = \star$, $m \leqslant n$.) The basic motivation for solving the above problem is given below. Given a collection of m machines it is clear that any other collection of size n for $n \geqslant m$ can simulate the m machines. Now is it possible to do so when $n < m$? If so, how efficiently can it be done? In this paper we will try to formalize the above idea in a very general way.

II. Patching and Amalgamation

Consider the following scenario. Given two IIMs, M_1 and M_2, we will construct another IIM M, that will simulate M_1 and M_2 using the following algorithm. M waits until the team outputs a program, say p_1, and then M outputs p_1. Then, if the team outputs another program p_2, M outputs a program which, when given an input, runs both p_1 and p_2 on that input in parallel until one of them gives an answer If both give an answer, then M chooses the output from the lexicographically least program. We will denote this program by $\{p_1, p_2\}$ and call the above program the *amalgamation* of p_1 and p_2. M also dovetails p_1 and p_2 on the input seen so far. The moment it finds a disagreement between the input seen so far and the results of the dovetail, it outputs a new guess: p_1, if the mistake was in p_2 and vice versa (i.e., M has eliminated the wrong program). We have just shown that $C(2, EX_0^0) \subseteq C(1, EX_2^0)$. The amalgamation technique was introduced in (Case and Smith, 1983) and also used in (Pitt, 1984; Pitt and Smith, 1988; Valiant, 1984).

We use the concept of *patching* to show that $C(2, EX_0^1) \subseteq C(1, EX_3^1)$. Again we will construct an M to simulate M_1 and M_2. As in the previous case, suppose M_1 and M_2 output p_1 and then p_2. M outputs p_1 and then $\{p_1, p_2\}$. M then looks for a mistake in the amalgamation by dovetailing p_1 and p_2 on the inputs seen so far. If it never finds a mistake, then $\{p_1, p_2\}$ is the correct program. If it finds a mistake, it remembers where the mistake occurred and outputs a new guess, which is a essentially $\{p_1, p_2\}$ patched with a table containing the correct value where the mistake occurred. We will denote this program by $\{p_1, p_2\}^1$. Now the machine looks to eliminate one of the programs p_1, p_2 by looking for two mistakes in any one of the programs, remembering the previous mistake. Suppose it does not find two additional mistakes. Then M has found at most one anomaly, for a total of two including the previous one which it has patched. Hence in this case the simulation works. If an additional mistake is found a new guess is produced. This guess is either p_1 or p_2, the one which did not have the two errors. Clearly $C(2, EX_0^1) \subseteq C(1, EX_3^1)$.

LEMMA 1. *Suppose an* IIM M *is simulating* m IIMs $M_1, M_2, ..., M_m$. *Assume the* $M_1, M_2, ..., M_m$ *are allowed only one guess, then the worst case number of mind changes for* M *occurs when*

1. Each M_i $(1 \leqslant i \leqslant m)$ *outputs a program.*

2. *The errors in any program which are output by* $M_1, M_2, ..., M_m$ *are discovered by* M *only after each* M_i *has output its program.*

Proof. We will prove this lemma for the $m = 3$ case only. The general case easily follows. Assume the second condition of the lemma is violated by the team. Suppose the team outputs a guess p_1. Since M is assumed to be performing an accurate simulation, it outputs p_1 since this could be the only guess output by the team. Suppose the machine M eliminates the guess p_1 by discovering enough errors before the next guess is output by the team. Then M can output the program p_2 which is the next guess by the team. Now the machine M again eliminates the guess p_2, by discovering enough errors before the next guess is output by the team. Then M waits for the last program p_3, to be output by the team and then outputs p_3. Since the first two guesses by the team were incorrect, the third guess must be the correct one. Hence M simulates the team with three guesses. But if the second condition in the lemma was not violated by the team, then before M could discard p_1, the team outputs its second guess. Now M has to output $\{p_1, p_2\}$, since it does not know which one of the guesses is the correct one. Now M dovetails $\{p_1, p_2\}$ on the inputs seen so far, so that it can eliminate the wrong program. However, again by the second condition of the lemma, if the third guess p_3 is output by the team before M eliminates any guesses, then M must amalgamate p_3 with p_1 and p_2; i.e., M must output $\{p_1, p_2, p_3\}$. Now M can dovetail $\{p_1, p_2, p_3\}$ on the input seen so far and discover errors in one program and output a new guess, eliminating the wrong one from the amalgamation. We will denote this program by $(\{p_1, p_2, p_3\})$. Similarly M can eliminate another program from the amalgamation and output $((\{p_1, p_2, p_3\}))$. If the second condition of the lemma is not violated the machine M needs five guesses. Hence if the second condition is not violated M needs more guesses. Also, clearly if the first condition in the lemma is not met M makes fewer guesses. Hence the first and the second condition should be satisfied for M to have the maximum number of mind changes. ∎

There are several different conditions which will lead to the worst case mind change for M. Note that the second condition in the lemma can be changed to "The errors in any program which are output by $M_1, ..., M_m$ are discovered by M only after *at least one more* M_i $(1 \leqslant i \leqslant m)$ has output its program" and still cause M to have the maximum number of mind changes. Using the above example M will output p_1, then $\{p_1, p_2\}$. Now

it can eliminate one program from the amalgamation and output $(\{p_1, p_2\})$. M then amalgamates the teams last guess p_3 with $(\{p_1, p_2\})$ and outputs a new guess $\{(\{p_1, p_2\}), p_3\}$. Observe that there are only two programs in the amalgamation. So M can eliminate one more from the amalgamation by dovetail on the input seen so far to output a new guess $(\{(\{p_1, p_2\}), p_3\})$. Hence M needs five guesses in all to simulate the team. Which is the same as in the case of the lemma. However, without loss of generality the proofs of all the results given below, we will assume the conditions given in the above lemma. The above assumption makes the proofs easier.

III. Simulating a Team with One Conjecture Using a Single IIM

THEOREM 2. $(\forall m > 1)(\forall a, b \in \mathbb{N})[b < 2(m-1) \Rightarrow C(m, EX_0^0) - C(1, EX_b^a) \neq \varnothing]$.

Proof. Let $m > 1$ and $a, b \in \mathbb{N}$ and M be any IIM. Assume $b < 2(m-1)$. We will construct a set S such that $S \in C(m, EX_0^0) - C(1, EX_b^a)$. The following is an intuitive description of S, a set of recursive functions. Given a $f \in S$ there will be at most $m-1$ other recursive functions in S which will have some same initial segment as f. In the longest such common initial segment there will be at most m special integers one of which will describe f. We will choose the special integers to be odd integers. Formally, $f \in S$ if and only if the following two conditions hold:

1. $f(x)$ is odd for no more than m distinct values of x.
2. There exists x and j such that $f(x) = 2j+1$ and $\varphi_j = f$.

For any $f \in S$, the ith member of the team on input f executes the following algorithm.

Begin M_i
1. Wait for the ith odd integer in the range of f, say k.
2. Output $(k-1)/2$.

End M_i

By the definition of S, one of the m odd integers in the range of f will describe f, therefore $S \in C(m, EX_0^0)$. We will construct an $f \in S$ such that, $f \notin EX_b^a(M)$. To construct f we will use a n-ary recursion theorem. The following is an intuitive description of the theorem. We can construct n programs $e_1, \ldots, e_n$, such that any program e_i in the sequence can use every program, including itself, in the sequence as an explicit extra parameter. This recursion theorem is a special case of the operator recursion theorem

(Case, 1974). Given a program $\alpha_{i,j}$, we will abbreviate $\varphi_{\alpha_{i,j}}$ by $f_{i,j}$. Using a $m(m+1)/2$-ary recursion theorem and a finite extension argument we construct functions $f_{1,1}, f_{2,1}, f_{2,2}, f_{3,1}, f_{3,2}, f_{3,3}, ..., f_{m,1}, f_{m,2}, ..., f_{m,m}$. For $i \in \{1, ..., m\}$ and $j \in \{1, ..., i\}$, $\sigma_{i,j}$ will denote the largest initial segment of $f_{i,j}$ constructed so far.

Phase 0. Initialize. Set $\sigma_{i,j} = \varnothing$. Let $x_{i,j}$ denote the largest value for which $\sigma_{i,j}$ is defined so far. Set $x_{1,1} = 0$ and $\sigma_{1,1}(x_{1,1}) = 2\alpha_{1,1} + 1$. Extend $\sigma_{1,1}$ with more and more values of 0's until $M(\sigma_{1,1})$ is defined, say $M(\sigma_{1,1}) = q$. If $M(\sigma_{1,1})$ is never defined, then $\sigma_{1,1}$ becomes a total function. Then, $f = \sigma_{1,1}$ is the desired function. We are mostly interested in initial segments on which M changes its mind. We will use variables $j_1, j_2, ..., j_m$ to keep track of on which initial segments M changes its mind. Since the first program was produced on input $\sigma_{1,1}$, set $j_1 = 1$.

The following algorithm, executed in m stages, will yield an f that is desired. Each stage is divided into two phases. If some phase does not terminate then we would have the desired f.

Phase 1. In this phase we will construct m identical initial segments such that M on that segment outputs m programs.

For $i = 2$ *to* m *do Begin step* i

1. Set $\sigma_{i,j} = \sigma_{i-1,1}$ for $j = 1$ to i (initialize $\sigma_{i,j}$)
   $\sigma_{i,j}(x_{i,j} + 1 = 2\alpha_{i,j} + 1$ for $j = 1$ to i (place a special integer)
2. Now simultaneously execute steps (i) and (ii) below until step (ii) is satisfied.
   (i) For $j \in \{1, ..., i\}$ simultaneously extend $\sigma_{i,j}$ with more and more values of $2j - 2$ (Each of the $\sigma_{i,j}$ start with the same initial segment and branch out differently.)
   (ii) Look for the least $l \in \{1, ..., i\}$ such that $M(\sigma_{i,l}) \neq q$ where $q = M(\sigma_{i-1,j_{i-1}})$. If such a j is found, let j_i be that j. Set $q = M(\sigma_{i,j_i})$.
3. Set $\sigma_{k,j_k} = \sigma_{i,j_i}$ for $k = 1, ..., i-1$. (Update all the previous segments on which M changed its mind. Now we have i identical segments on which M has output i programs.)

For any i, M changes its mind on σ_{i,j_i} (from $\sigma_{i-1,j_{i-1}}$). Otherwise there exists an i for which clause (ii) in step (2) is not satisfied. Then by clause (i) in step (2) the functions $f_{i,1}, ..., f_{i,i}$ are all total and pairwise infinitely different. Also $M(f_{i,j})\downarrow = q$ for $1 \leqslant j \leqslant i$. Since $i \geqslant 2$ it cannot be the case that $\varphi_q = f_{i,1}$ and $\varphi_q = f_{i,2}$. (This technique of finding more than one function that causes the inference machine to converge to the same program was originally used in Barzdin, 1974. This technique is used extensively below.) There are exactly i odd integers in the range of $f_{i,1}$ and $f_{i,2}$. By construction of the above functions they both belong to S. Hence the desired f would be either $f_{i,1}$ or $f_{i,2}$.

End step i

Note that for each i, if and when the above loop terminates, $j_1, ..., j_m$ are now defined. $\sigma_{1,j_1}, \sigma_{2,j_2}, ..., \sigma_{m,j_m}$ will all have the same initial segment on which M outputs m programs, the last of which is q. Also each initial segment contains exactly m special integers. Let $q = M(\sigma_{1,j_1})$.

Phase 2. Using the m initial segments obtain $m-1$ mind changes by diagonalization.

For $i=1$ *to* $m-1$ *do Begin step* i

1. Set $x = x_{i,j_i}$
2. Simultaneously execute steps (a), (b), and (c) for $s = 1, 2, \ldots$ until condition (B) or (C) is satisfied.
   (a) Set $\sigma_{i,j_i}(x_{i,j_i}+1)=0$
   (b) Look for distinct points $y_0, \ldots, y_a$ in the set $\{x+1, \ldots, x+s\}$, such that φ_q is convergent on those points.
   (c) See if $M(\sigma_{i,j_i}) \neq q$.

Condition (B). There exists $y_0, \ldots, y_a$ such that $\forall y \in \{y_0, \ldots, y_a\}$, $\varphi_q(y)\downarrow$. Then set

$$\sigma_{i+1,j_{i+1}}(z) = \begin{cases} 2 \dot{-} 2\varphi_q(z), & \text{for } z \in \{y_0, \ldots, y_a\}; \\ \sigma_{i,j_i}(z), & \text{for } x_{i+1,j_{i+1}} \leqslant z \leqslant x_{i,j_i} \wedge z \notin \{y_0, \ldots, y_a\}. \end{cases}$$

Extend $\sigma_{i+1,j_{i+1}}$ with more and more values of 0's until $M(\sigma_{i+1,j_{i+1}}) \neq q$. If such an extension is not found then $f_{i+1,j_{i+1}}$ is a total recursive function and $M(f_{i+1,j_{i+1}})\downarrow = q$. By the construction of $\sigma_{i+1,j_{i+1}}$ above, $f_{i+1,j_{i+1}} \in S$ and $\text{card}(\{z \mid \varphi_q(x) \neq f_{i+1,j_{i+1}}(z)\}) > a$ hence the desired f would be $f_{i+1,j_{i+1}}$.

Condition (C). If $M(\sigma_{i,j_i}) \neq q$, then set $\sigma_{i+1,j_{i+1}} = \sigma_{i,j_i}$.

For each i the above loop terminates. Otherwise there exists an i for which condition (B) and (C) in step (2) is not satisfied. By clause (a) in step (2) f_{i,j_i} is a total function. By clause (c) in step (2), $M(f_{i,j_i})\downarrow = q$, and by clause (b) program q computes a finite function. Hence $f_{i,j_i} \notin EX_b^a(M)$. By the construction of σ_{i,j_i}, $f_{i,j_i} \in S$, therefore the desired f will be f_{i,j_i}.

End step i

Clearly if the above algorithm does not terminate, by the construction there will an f (one of the $f_{i,j}$'s) such that $f \in S$ and $f \notin EX_b^a(M)$. Suppose the above loop terminates, then M on σ_{m,j_m} will output $2m-1$ programs. Let $f = \sigma_{m,j_m} \cup \{(x, 0) \mid x > x_{m,j_m}\}$ and let program coded by α_{m,j_m} compute f. Clearly $f \in S$ and $f \notin EX_b^a(M)$. ∎

LEMMA 3. $(\forall m > 1)(\forall a, c \in \mathbb{N})[c \geqslant (a+2)(m-1) \Rightarrow C(m, EX_0^a) \subseteq C(1, EX_c^a)]$.

Proof. Let $m > 1$, $a \in \mathbb{N}$, and $M_1, \ldots, M_m$ be a team of IIM's. Assume the team outputs programs $p_1, p_2, \ldots, p_m$ in that order. Now we will describe a IIM M to simulate the team of m machines. M outputs p_1, $\{p_1, p_2\}$, $\{p_1, p_2, p_3\}, \ldots, \{p_1, p_2, \ldots, p_m\}$. Note that by Lemma 1 this leads to the worst case mind changes for the machine M. Now the machine M looks to eliminate programs from the amalgamation $\{p_1, \ldots, p_m\}$. M dovetails each of the programs $p_1, \ldots, p_m$ on the input seen so far. If it discovers $a+1$ errors in any one of the p_i's it eliminates the p_i from the amalgamation. However, it could be the case that all the programs output by the team can have at most a errors. In which case M cannot eliminate any of the programs. If the errors in each program do not overlap and if all the errors

are convergent errors, then the amalgamation will have am errors in it. So, if M has to simulate the team with at most a anomalies, this cannot be its last guess. Hence, while M is in the process of looking for $a+1$ errors, it outputs a new guess whenever it sees an error in any program p_i. This new guess is essentially the same as the last guess with a table containing the correct value for the error in p_i. M also remembers on which program the error occurred, so that if a program has more than a errors it can be eliminated from the amalgamation. The above procedure is repeated for every error M encounters. In the worst case, since there are m programs and the number of errors allowed is a, M has to make $a(m-1)$ patches. That is, it has to output $a(m-1)$ programs for patching. Now after $a(m-1)$ patches M could discover $a+1$ errors in $m-1$ programs one at a time. In which case it has to eliminate the incorrect program from the amalgamation, one at a time; i.e., it has to output $m-1$ new guesses to get the final correct guess. In all, M needs $a(m-1)+m+m-1$ programs, i.e., it needs $(a+2)(m-1)$ mind changes. ∎

THEOREM 4. $(\forall m>1)(\forall a, c\in\mathbb{N})[C(m, EX_0^a)\subseteq C(1, EX_c^b)$ *iff* $c\geqslant 2(m-1)$ *and* $b\geqslant a+\lfloor a(m-1)/(c-(2m-3))\rfloor]$.

Proof. ($\Leftarrow$) Let $m>1$, $a, c\in\mathbb{N}$, and $M_1, \ldots, M_m$ be IIMs. Assume $c\geqslant 2(m-1)$ and $b\geqslant a+\lfloor a(m-1)/(c-(2m-3))\rfloor$. Let $L=a(m-1)/(c-(2m-3))$. We will construct a IIM M such that it will simulate the collection of m machines. For M to output the maximum number of guesses, by Lemma 1, the collection of m machines outputs m different programs. Without loss of generality let $p_1, p_2, \ldots, p_m$ be the programs output in that order. M will use the following algorithm to simulate the team.

Set $count=0$. On input $f(x)$, M executes the following steps in order. Initially, M outputs the first program produced by the team. If more than one guess is produced simultaneously, chose the lexicographically least one.

Begin M

(1) Let p be the last guess output by M. If the team output a new guess q on the input $f(x)$, then output $\{q, p\}$.

(2) Run all the programs output by the team on the input seen so far for $\leqslant x$ steps.

(3) Eliminate programs which have greater than a errors from the amalgamation (elimination is a new guess) using results in step (2).

(4) If there are at least $\lceil L\rceil$ errors in the last program output by M and **if** $count<c-(2m-2)$ **then**

  (a) Patch the amalgamation (if M patches, then it outputs a new guess).

  (b) $count=count+1$

  (c) Remember where the errors occurred, to be used in step (3).

End M

By Lemma 1, for M to have the maximum number of mind changes, no errors can be discovered before the team has output all its m guesses. Hence M by step (1) outputs p_1, $\{p_1, p_2\}$, ..., $\{p_1, p_2, ..., p_m\}$ before it starts patching. Now the machine M in step (4) is looking for $\lceil L \rceil$ errors to patch. Since $c \geq 2(m-1)$, M uses $c-(2m-2)$ guesses to do the patching. Hence, the total number of patches possible is $(c-(2m-2))\lceil L \rceil$. We will show that M simulates the team correctly, i.e., $C(m, EX_0^a) \subseteq (1, EX_c^b)$.

Case (1) All the programs in the amalgamation are correct. The maximum number of errors in the amalgamation is ma. This occurs when all the programs output by the m machines are correct with a convergent errors, and the errors in one program do not overlap with errors of other programs. Hence, none of the programs output by the team can be eliminated from the amalgamation. M finds $(c-(2m-2))\lceil L \rceil$ errors and patches them; in the worst case the total number of errors left is $ma-(c-(2m-2))\lceil L \rceil$. Now $ma-(c-(2m-2))\lceil L \rceil = ma-(c-(2m-3))\lceil L \rceil + \lceil L \rceil$. Suppose L is integer, then $ma-(c-(2m-2))\lceil L \rceil = a+L$. If L is not an integer, then $ma-(c-(2m-2))\lceil L \rceil \leq a+\lfloor L \rfloor$.

Case (2) Some of the programs in the amalgamation are correct. If M discovers all the incorrect programs it will eliminate them. Therefore M simulates the team correctly. If it cannot discover any of the errors, then the errors are errors of omission. The errors of omission in the amalgamation do not create additional errors. Suppose M uses all its $c-(2m-2)$ guesses to patch. Then by Case (1) the remaining number of errors in the amalgamation is less than or equal to $a+\lfloor L \rfloor$. Suppose M has not used all its $c-2m+2$ guesses to patch, then it has not found $\lceil L \rceil$ errors in the amalgamation. That is, it has discovered at most $\lfloor L \rfloor$ errors. It is also possible for the correct program to have a total of a errors of omission. The rest of the programs in the amalgamation can have a errors of omission at the same place as the correct one. Hence in the worst case, the maximum errors possible is $a+\lfloor a(m-1)/(c-(2m-3)) \rfloor$. This proves the "only if" part.

($\Rightarrow$) Let $m>1$, $a, c \in \mathbb{N}$, and $M_1, ..., M_m$ be IIMs. Let M be any IIM. Suppose by way of contradiction that $C(m, EX_0^a) \subseteq C(1, EX_c^b)$ and either $c<2(m-1)$ or $b<a+\lfloor a(m-1)/(c-(2m-3)) \rfloor$.

If $c<2(m-1)$ and $C(m, EX_0^a) \subseteq C(1, EX_c^b)$. Then by Theorem 2 $(\forall b \in \mathbb{N})$ $C(m, EX_0^0) - C(1, EX_c^b) \neq \varnothing$. Therefore $C(m, EX_0^a) - C(1, EX_c^b) \neq \varnothing$. This is a contradiction, hence $C(m, EX_0^a) \subseteq C(1, EX_c^b)$ implies $c \geq 2(m-1)$. Suppose $c \geq 2(m-1)$ and $b<a+\lfloor a(m-1)/(c-(2m-3)) \rfloor$.

We will define a set S such that $S \in C(m, EX_0^a) - C(EX_c^b)$. The following is an intuitive description of S, a set of recursive functions. Given a $f \in S$ there will be at most $m-1$ other recursive functions in S which will have

the same initial segment as f. In that common initial segment there will be at most m special integers one of which will describe an a variant of f. We will choose the special integers to be odd integers. Formally, $f \in S$ if and only if the following two conditions hold:

1. $f(x)$ is odd for no more than m distinct values of x.
2. There exists x and j such that $f(x) = 2j + 1$ and $\varphi_j =^a f$.

For any $f \in S$, the ith member of the team on input f executes the following algorithm.

Begin M_i
1. Wait for the ith odd integer, say k.
2. Output $(k-1)/2$.

End M_i

By the definition of S, for any $f \in S$ one of the m integers in f will describe it except at most at a anomalous inputs. Therefore $S \in C(m, EX_0^a)$. We will construct an f such that, $f \in S$ and $f \notin EX_c^b(M)$. Given a program $\alpha_{j,k}$, we will abbreviate $\varphi_{a,k}$ by $f_{j,k}$. Using a $m(m+1)/2$-ary recursion theorem and a finite extension argument we construct functions $f_{j,k}$ for $j \in \{1, ..., m\}$, and $k \in \{1, ..., j\}$. For $j \in \{1, ..., m\}$, and $k \in \{1, ..., j\}$ let $\sigma_{j,k}$ denote the largest initial segment of $f_{j,k}$ constructed so far. Let $x_{j,k}$ denote the largest value in the domain of $\sigma_{j,k}$.

Phase 0. Initialize. Set $x_{1,1} = 0$, $\sigma_{j,k} = \varnothing$, and $\sigma_{1,1}(x_{1,1}) = 2\alpha_{1,1} + 1$. Extend $\sigma_{1,1}$ with more and more values of 0's until $M(\sigma_{1,1})$ is defined, $M(\sigma_{1,1}) = q$. If $M(\sigma_{1,1})$ is never defined then $\sigma_{1,1}$ becomes a total function which will serve as our f. Throughout this proof q will denote the current guess of M on the f defined so far. We are mostly interested in initial segments on which M changes its mind. We use variables $k_1, k_2, ..., k_m$ to keep track of on which initial segments M changes its mind. Since the first mind change was on $\sigma_{1,1}$, we set $k_1 = 1$ (note that M has output one program).

The following algorithm when executed will yield the desired f. The algorithm is divided into 3 phases. If some phase fails to terminate we would have the desired f.

Phase 1. Construct m identical initial segments such that M on that segment outputs m programs.

For $j = 2$ *to* m *do Begin step* j
1. Set $\sigma_{j,k} = \sigma_{j-1,1}$ for $k = 1$ to j (initialize $\sigma_{j,k}$)
   $\sigma_{j,k}(x_{j,k} + 1) = 2\alpha_{j,k} + 1$ for $k = 1$ to j (place special integers)
2. Now simultaneously execute step (i) and (ii) below until condition (II) is satisfied.
   (i) For $k \in \{1, ..., j\}$ simultaneously extend $\sigma_{j,k}$ with more and more values of $2k - 2$ (each $\sigma_{j,k}$ start with the same initial segment and branch out differently)

(ii) Look for the least $k \in \{1, ..., j\}$ such that $M(\sigma_{j,k}) \neq q$, where $q = M(\sigma_{j-k,k_{j-1}})$.

3. Set $\sigma_{l,k_l} = \sigma_{j,k_j}$ for $l = 1, ..., j-1$ (update all the previous segments on which M changed its mind. Now we have j identical segments on which M has changed its mind j times)

Condition (II). If there exists $k \in \{1, ..., j\}$ such that $M(\sigma_{j,k}) \neq q$. Let k_j be least such a k and set $q = M(\sigma_{j,k_j})$.

For each j, M changes its mind on σ_{j,k_j}. Otherwise there exists an i and a j for which condition (II) is not satisfied. Then by clause (i) in step (2) the functions $f_{j,1}, ..., f_{j,j}$ are total and pairwise infinitely different. Also by clause (ii) in step (2) $M(f_{j,k})\downarrow = q$ for $1 \leqslant k \leqslant j$. By the construction of the above functions they each belong to S. Since $j \geqslant 2$ it cannot be the case that $\varphi_q = {}^b f_{j,1}$ and $\varphi_q = {}^b f_{j,2}$. Hence the desired f would be either $f_{j,1}$ or $f_{j,2}$.

End step j.

Note that for each i, if and when the above loop terminates, $k_1, ..., k_m$ are defined. $\sigma_{1,k_1}, \sigma_{2,k_2}, ..., \sigma_{m,k_m}$ all have the same initial segment on which M has output m programs. Also each initial segment contains exactly m special integers.

Phase 2. Use the m identical segments, diagonalization techniques, and anomaly markers to obtain mind changes.

We will obtain $c + 3 - 2m$ mind changes for M using diagonalization and by placing anomaly markers in the domain of σ_{m,k_m}. Let $q = M(\sigma_{m,k_m})$. First consider the case when $\lfloor a(m-1)/(c+3-2m) \rfloor \leqslant a$. We will diagonalize against the current guess of M, using the segments $\sigma_{1,k_1}, ..., \sigma_{m-1,k_{m-1}}$. Also, we make sure the number of points at which σ_{j,k_j} (for $j < m$), is different from σ_{m,k_m} is not greater than a. We use the variable *Diag* to keep track of the number of errors on each segment. The variable t is used to keep track of the number of differences between φ_q and σ_{m,k_m}. When t reaches the value $\lfloor a(m-1)/(c+3-2m) \rfloor$, using a anomaly markers we force M to change its mind. Set $t = 0$.

For $j = 1$ *to* $m-1$ *begin step j*

($\mathscr{A}$) Set $Diag = 0$

($\mathscr{B}$) **While** $Diag < a$ **do**

Execute step (a) and (b) simultaneously until a point y_t is found in step (b), and then the rest in order.

(a) Extend σ_{j,k_j} with 0's.

(b) Dovetail φ_q on all the points in the extension made to σ_{j,k_j} until a convergent point y_t is found.

If program q does not halt, then it is computing a finite function. The function f_{j,k_j} is a total function. Let $f = \sigma_{m,k_m} \cup \{(x, f_{j,k_j}(x)) \mid x > x_{m,k_m}\}$. Then $M(f)\downarrow = q$. Therefore $f \notin EX_c^b(M)$. By construction of σ_{j,k_j}, $f \in S$, hence we have the desired f.

(c) Set $Diag = Diag + 1$ and $t = t + 1$

(d) **If** $t = \lfloor a(m-1)/(c+3-2m) \rfloor$ **then do**

(1) Set

$$\sigma_{m,k_m}(x) = \begin{cases} 2 \dot{-} 2\varphi_q(x), & \text{if } x \in \{y_0, ..., y_{t-1}\}; \\ \sigma_{j,k_j}(x), & \text{if } x_{m,k_m} + 1 \leqslant x \leqslant x_{j,k_j} \text{ and } x \notin \{y_0, ..., y_{t-1}\}. \end{cases}$$

(2) Set $t=0$

Place a anomaly markers $t_1, ..., t_a$ on consecutive values, starting from the least point not yet in the domain of σ_{m,k_m}. Since σ_{m,k_m} is defined $\forall x \leqslant x_{m,k_m}$, the anomaly markers will be placed at points $x_{m,k_m}+1, ..., x_{m,k_m}+a$. Execute the steps 3, 4, and 5 simultaneously until condition (IV) or (V) is satisfied.

(3) Extend σ_{m,k_m} with more and more values of 0's except at the points where the anomaly markers are kept.

(4) Dovetail φ_q at the points $t_1, ..., t_a$.

(5) Let $\tau = \sigma_{m,k_m} \cup \{(x,0) | t_1 \leqslant x \leqslant t_a\}$. See if $M(\tau) \neq q$.

Condition (IV). If $\varphi_q(x)\downarrow$ for all x such that $t_1 \leqslant x \leqslant t_a$. Then set $\sigma_{m,k_m}(x) = 2 \dot{-} 2\varphi_q(x)$ for $t_1 \leqslant x \leqslant t_a$. Extend σ_{m,k_m} with more and more values of 0's until $M(\sigma_{m,k_m}) \neq q$. If M does not chage its mind, on some extension of σ_{m,k_m} then f_{m,k_m} is a total function and $M(f_{m,k_m})\downarrow = q$. Clearly $f_{m,k_m} \in S$, and f_{m,k_m} differs at least at $a + \lfloor a(m-1)/(c+3-2m) \rfloor$ points from φ_q. Hence $f_{m,k_m} \notin EX_c^b(M)$. Hence the desired f is f_{m,k_m}. Suppose M changes its mind on some extension of σ_{m,k_m} then set $q = M(\sigma_{m,k_m})$. Extend $\sigma_{1,k_1}, ..., \sigma_{m-1,k_{m-1}}$ to match σ_{m,k_m}.

Condition (V). If $M(\tau) \neq q$ then set $q = M(\tau)$ and set $\sigma_{m,k_m}(x) = 0$ for $t_1 \leqslant x \leqslant t_a$. Extend $\sigma_{1,k_1}, ..., \sigma_{m-1,k_{m-1}}$ to match σ_{m,k_m}. Note that if condition (IV) is satisfied then σ_{1,k_1} differs from σ_{m,k_m} at most at $\lfloor a(m-1)/(c+3-2m) \rfloor$ points.

If conditions (IV) and (V) are never satisfied, then f_{m,k_m} is defined everywhere except at the points $t_1, ..., t_a$. Since condition (IV) was not satisfied there exists some anomaly markers $\{t_{i_1}, ..., t_{i_r}\} \subseteq \{t_1, ..., t_a\}$ for $r \leqslant a$ for which program q does not halt. Hence φ_q is undefined at points $t_{i_1}, ..., t_{i_r}$. Let f be the function defined as follows:

$$f(x) = \begin{cases} f_{m,k_m}(x), & \text{if } x \notin \{t_1, ..., t_a\}; \\ 2 \dot{-} 2\varphi_q(x), & \text{if } x \notin \{t_{i_1}, ..., t_{i_r}\}; \\ 0, & \text{otherwise.} \end{cases}$$

Now $M(f)\downarrow = q$ and $f = {}^a f_{m,k_m}$. Hence $f \in S$, but $\text{card}(\{x | \varphi_q(x) \neq f(x)\}) > b$, therefore $f \notin EX_c^b(M)$. If condition (IV) or (V) is satisfied, we can force M to change it mind again.

End if

End while

(*G*) Set

$$\sigma_{j+1,k_{j+1}}(x) = \begin{cases} 2 \dot{-} 2\varphi_q(x), & \text{if } x \in \{y_0, ..., y_{t-1}\}; \\ \sigma_{j,k_j}(x), & \text{if } x_{j+1,k_{j+1}} < x \leqslant x_{j,k_j} \text{ and } x \notin \{y_0, ..., y_{t-1}\} \end{cases}$$

End step j

If Phase 2 is not completed then we would have the desired f. Suppose Phase 2 is completed. Then M has changed its mind $a(m-1)/\lfloor a(m-1)/(c+3-2m) \rfloor$ times, i.e., at least $c+3-2m$ times on the segment σ_{m,k_m}. Now if $\lfloor a(m-1)/(c+3-2m) \rfloor > a$ then the errors are distributed to more than one initial segment. Hence each σ_{j,k_j} for $j = 1, ..., m-1$ varies from σ_{m,k_m} at most at a points at the end of this Phase. Let $q = M(\sigma_{m,k_m})$.

Phase 3. Using the initial segments force M to change its mind $m-1$ times by diagonalization.

For $j=1$ *to* $m-1$ *do begin step* j

1. Set $x = x_{j,k_j}$.
2. Simultaneously execute steps (a), (b), and (c) for $s = 1, 2, \ldots$ until condition (B) or (C) is satisfied.
   (a) Set $\sigma_{j,k_j}(x+s) = 0$
   (b) Look for $b+1$ distinct points in the set $\{x+1, \ldots, x+s\}$ on which φ_q is convergent.
   (c) See if $M(\sigma_{m,k_m} \cup (\sigma_{j,k_j} - \sigma_{m,k_m})) \neq q$.

Condition (B). If there exists $y_0, \ldots, y_b$ such that $\forall y \in \{y_0, \ldots, y_b\}\ \varphi_q(y)\downarrow$.
Then set

$$\sigma_{m,k_m}(z) = \begin{cases} 2 \dot{-} 2\varphi_q(z), & \text{if } z \in \{y_0, \ldots, y_b\}; \\ \sigma_{j,k_j}(z), & \text{if } x_{j+1,k_{j+1}} \leqslant z \leqslant x_{j,k_j} \wedge z \notin \{y_0, \ldots, y_b\}. \end{cases}$$

Extend σ_{m,k_m} with more and more values of 0's until $M(\sigma_{m,k_m}) \neq q$. If such an extension is not found then f_{m,k_m} is a total recursive and $M(f_{m,k_m})\downarrow = q$. By the construction of σ_{m,k_m} above, $f_{m,k_m} \in S$ and $\operatorname{card}(\{x \mid \varphi_q \neq f_{m,k_m}\}) > b$, hence $f_{m,k_m} \notin EX_c^b(M)$. Therefore the desired f would be f_{m,k_m}. Suppose M changes its mind on some extension of σ_{m,k_m}, let $q = M(\sigma_{m,k_m})$. For $j+1 \leqslant l \leqslant m-1$ set

$$\sigma_{l,k_l}(z) = \sigma_{m,k_m}(z) \qquad \text{for} \quad x_{l,k_l} \leqslant z \leqslant x_{m,k_m}.$$

Condition (C). If $M(\sigma_{m,k_m} \cup (\sigma_{j,k_j} - \sigma_{m,k_m})) \neq q$. Then set $q = M(\sigma_{m,k_m} \cup (\sigma_{j,k_j} - \sigma_{m,k_m}))$. For $j+1 \leqslant l \leqslant m$ set

$$\sigma_{l,k_l}(z) = \sigma_{l,k_l}(z) \qquad \text{for} \quad x_{l,k_l} \leqslant z \leqslant x_{j,k_j}.$$

For each j the above loop terminates. Otherwise there exists a j for which conditions (B) and (C) are not satisfied. By clause (a) in step (2) f_{j,k_j} is a total function. Let $f = \sigma_{m,k_m} \cup (f_{j,k_j} - \sigma_{m,k_m})$. By clause (c) in step (2), $M(f)\downarrow = q$, and by clause (b) program q computes a finite function. Hence $f \notin EX_c^b(M)$, by construction of $\sigma_{j,k_j} f \in S$; therefore we have the desired f. Hence for each j, M is forced to change its mind.

End step j

Clearly, if the algorithm does not terminate, by the construction there will be an f such that $f \in S$ and $f \notin EX_c^b(M)$. Now if the above algorithm terminates, then M on σ_{m,k_m} will output $m + c + 3 - 2m + m - 1$ programs. Hence M makes $c+1$ mind changes. This contradicts the assumption that M only makes c mind changes. Let $f = \sigma_{m,k_m} \cup \{(x, 0) \mid x > x_{m,k_m}\}$. Clearly $f \in S$ and $f \notin EX_b^c(M)$ for every $b < a + \lfloor a(m-1)/(c+3-2m) \rfloor$. ∎

The following lemma is a consequence of Theorem 2.6 of (Case and Smith, 1983).

LEMMA 5. $(\forall m > 1)$ *and* $(\forall a \in \mathbb{N})$ *if* $b < a$ *then* $[C(m, EX_0^a) - C(1, EX_\star^b) \neq \varnothing$.

In all the results obtained so far, we derived necessary and sufficient conditions, for a single machine to simulate a team of machines, with each member of the team allowed one guess. In the rest of this section we will derive necessary and sufficient conditions for a single machine to simulate a team of machines, with each member of the team allowed a finite number of guesses.

LEMMA 6. *Suppose an* IIM M *is simulating a team of* m IIMs $M_1, M_2, ..., M_m$. *Assume each member of the team is allowed at most* a *guesses, for some* $a \in \mathbb{N}$. *Then the worst number of mind changes for* M *occurs when the following two condition hold.*

1. *Each* M_j $(1 \leqslant j \leqslant m)$ *outputs* a *programs.*
2. *Let the* i*th* $\leqslant a$ *guess of the* M_j*th machine be* $p_{j,i}$. *Suppose* M_j *for* $1 \leqslant j \leqslant m$ *has outputs its* i*th guess, then the next guess by machine* M_j *can only be output, if* M *has eliminated* $p_{j,i}$ *from its amalgamation.*

Proof. We will prove this lemma for the case $m = 3$ and $a = 2$. The proof of the general case is similar. This proof is a generalization of the proof of Lemma 1. If any member of the team fails to output all of its allotment of guesses, then any IIM simulating the team will have fewer programs to consider, making its job easier. Hence, condition (1) is satisfied. Suppose the team initially output programs $p_{1,1}, p_{2,1}, p_{3,1}$ in that order, M outputs programs $p_{1,1}$, $\{p_{1,1}, p_{2,1}\}$, $\{p_{1,1}, p_{2,1}, p_{3,1}\}$. Suppose by way of contradiction that condition (2) is not satisfied. Then the team can output another guess, before any elimination. With out loss of generality, assume that M_1 outputs its next guess, $p_{1,2}$. Then immediately M changes its guess to $\{p_{1,2}, p_{2,1}, p_{3,1}\}$. Now if M discovered errors in the program $p_{1,1}$ before M_1 outputs a new guess, then M will first output $\{p_{2,1}, p_{3,1}\}$, then wait for M_1 to output its new guess and output $\{p_{1,2}, p_{2,1}, p_{3,1}\}$. Clearly condition (2) implies that any simulator of the team will output more conjectures. ∎

Now to make the proof of the following theorem easier, we will in addition to the two conditions given above make the following assumption. Any M_j can output its $(i+1)$th guess only if every other team member has output its ith guess and M has eliminated all the programs from the amalgamation except $p_{k,i}$ for $k \neq i$. Observe that making this assumption does not change the number of mind changes for M.

IV. Simulating a Team with a Single IIM

THEOREM 7. $(\forall m > 1)$ $(a, b, c \in \mathbb{N})$, $[C(m, EX_b^a) \subseteq C(1, EX_d^c)]$ *if and only if* $c \geqslant a$ *and*

$$d \geqslant \left\lfloor \left\lfloor \frac{a(m-1)}{c-a+1} \right\rfloor (b+1) + 2mb + \left\lfloor \frac{a}{c-a+1} \right\rfloor b + 2(m-1) \right\rfloor.$$

Proof. $(\Leftarrow)$ This proof shows that $\lfloor a(m-1)/(c-a+1) \rfloor (b+1) + 2mb + \lfloor a/(c-a+1) \rfloor b + 2(m-1)$ mind changes is sufficient for some M to simulate a team of m machines. Let $a, b, c \in \mathbb{N}$, and $M_1, M_2, ..., M_m$ be a team of IIMs. Assume $c \geqslant a$ and $d \geqslant \lfloor a(m-1)/(c-a+1) \rfloor (b+1) + 2mb + \lfloor a/(c-a+1) \rfloor b + 2(m-1) \rfloor$. The i th guess by the machine M_j will be denoted by $p_{j,i}$. We will construct a machine M which will simulate the team of m machines. By Lemma 6 for M to have the maximum number of mind changes, the team should output all the allotted $m(b+1)$ programs. In addition, each member of the team should only output its jth guess only after every other member has output their $(j-1)$th guess and M has eliminated all but one program from the amalgamation. M will simulate the team using the following algorithm. Suppose the team guesses programs $p_{1,1}, ..., p_{m,1}$. Assuming the worst case mild change for M by Lemma 6, it will output $p_{1,1}$, $\{p_{1,1}, p_{2,1}\}$, ..., $\{p_{1,1}, ..., p_{m,1}\}$ (M has output m guesses). Again by Lemma 6 and the assumption we made, for M to have the maximum number of mind changes, the team should not output any more guesses until M has eliminated all but one program from the amalgamation $\{p_{1,1}, ..., p_{m,1}\}$. Now M dovetails each $p_{i,1}$ for $1 \leqslant i \leqslant m$ on the input seen so far. It looks for $c-a+1$ errors in the amalgamation to patch. When M makes a patch it also remembers the programs that had the errors. M eliminates a program from the amalgamation when it discovers $a+1$ errors in that program. M uses $\lfloor a(m-1)/(c+1-a) \rfloor$ of its guesses to do the patching. For M to have the maximum number of mind changes, it has to use all $\lfloor a(m-1)/(c+1-a) \rfloor$ guesses to do the patching before it starts eliminating programs from the amalgamation (M has output $m + \lfloor a(m-1)/(c+1-a) \rfloor$ guesses). Now again by Lemma 6, M eliminates all but one program, $p_{i,1}$ for some $1 \leqslant i \leqslant m$ from its amalgamation (M has output $2m-1+\lfloor a(m-1)/(c+1-a) \rfloor$ guesses.

Suppose M_i is the member of the team which outputs the next guess, then M can immediately change its guess to what was output by M_i. But if the new guess is from a machine M_j for $i \neq j$, then M has to amalgamate $p_{i,1}$ and $p_{j,2}$ and output a new guess $\{p_{i,1}, p_{j,2}\}$ (M has output $2m + \lfloor a(m-1)/(c+1-a) \rfloor$ guesses). Now M starts to dovetail $\{p_{i,1}, p_{j,2}\}$ on the input seen so far and looks for $c+1-a$ errors and uses $\lfloor a/(c+1-a) \rfloor$ of its guesses to patch the program $\{p_{i,1}, p_{j,2}\}$ (M has output $2m + \lfloor a(m-1)/(c+1-a) \rfloor + \lfloor a/(c+1-a) \rfloor$ guesses). M also looks for

$a+1$ errors, in any one of the amalgamated programs, so that it can eliminate that program from its amalgamation. Again by Lemma 6 for the worst case number of mind changes, M should finish its patching before it eliminates $p_{i,1}$ (M has output $2m+\lfloor a(m-1)/(c+1-a)\rfloor+\lfloor a/(c+1-a)\rfloor+1$ guesses).

So far, a team member M_j has output two gesses $p_{j,1}$, $p_{j,2}$ and every member has output its first guesses. M has output a total of $2m+\lfloor a(m-1)/(c+1-a)\rfloor+\lfloor a/(c+1-a)\rfloor+1$ guesses and its current is $p_{j,2}$. Hence M needs $2m+\lfloor a(m-1)/(c+1-a)\rfloor+\lfloor a/(c+1-a)\rfloor$ guesses to eliminate the first guess of each member of the team. For M to have the maximum number of mind changes the team and M should repeat the above process. After each team member has output b of its allotted $b+1$ guesses, M would have output $2mb+\lfloor a(m-1)/(c+1-a)\rfloor b+\lfloor a/(c+1-a)\rfloor b$ guesses. When each team member outputs its last guess, M has to amalgamate (m guesses) and then patch ($\lfloor a(m-1)/(c+1-a)\rfloor$ guesses) and then eliminate all the programs except one ($m-1$ guesses). Hence, in all, M needs $2m-1+\lfloor a(m-1)/(c+1-a)\rfloor$ additional guesses. Totaling the above M needs $\lfloor a(m-1)/(c-a+1)\rfloor(b+1)+2mb+\lfloor a/(c-a+1)\rfloor b+2m-1$ guesses.

($\Rightarrow$) Let $m>1$ and $a, b, c\in\mathbb{N}$. If $c<a$ then by Lemma 5, $C(m, EX_b^a)-C(1, EX_\star^c)\neq\varnothing$ and the theorem follows. Hence, suppose $c\geqslant a$. Let M be any IIM. We will prove the contrapositive: if $d<\lfloor a(m-1)/(c-a+1)\rfloor(b+1)+2mb+\lfloor a/(c-a+1)\rfloor b+2(m-1)$ then $C(m, EX_b^a)-C(1, EX_d^c)\neq\varnothing$. The following is an intuitive description of S, a set of recursive functions. Given a $f\in S$ there will be at most $m-1$ other recursive functions in S which will have the same initial segment as f. In that common initial segment there will be at most $(b+1)m$ special integers one of which describe an a variant of f. We will choose the special integers to be odd integers. Formally, $f\in S$ if and only if the following two conditions hold.

1. $f(x)$ is odd for no more than $(b+1)m$ distinct values of x.

2. Suppose $\text{card}(\{x\mid f(x)\text{ is odd}\})\leqslant m$ then there exists x and j such that $f(x)=2j+1$ and $\varphi_j={}^a f$. Otherwise let x_1 be such that $\text{card}(\{x>x_1\mid f(x)\text{ is odd}\})=m$ then exists an $x>x_1$ and j such that $f(x)=2j+1$ and $\varphi_j={}^a f$.

For any $f\in S$, ith member of the team on input f executes the following algorithm.

Begin M_i
  For $j=0$ *to* b *do*
  (1) Wait for the $(jm+i)$th odd integer, say k.
  (2) Output $(k-1)/2$.
  End step j
End M_i

By the definition of S, for any $f \in S$ one of the last m integers in f will describe it except on at most a anomalous inputs. The team outputs the odd integers in a round robin manner. The last m guesses by the team are the last m odd integers, therefore $S \in C(m, EX_b^a)$. We will construct an f such that, $f \in S$ and $f \notin EX_d^c(M)$. Given a program $\alpha_{i,j,k}$, we will abbreviate $\varphi_{i,j,k}$ by $\varphi_{i,j,k}$. Using $a(b+1)m(m+1)/2$-ary recursion theorem and a finite extension argument we construct functions $f_{i,j,k}$ for $i \in \{0, ..., b\}$, $j \in \{1, ..., m\}$, and $k \in \{1, ..., j\}$. For $i \in \{0, ..., b\}$, $j \in \{1, ..., m\}$, and $k \in \{1, ..., j\}$ let $\sigma_{i,j,k}$ denote the largest initial segment of $f_{i,j,k}$ constructed so far. Let $x_{i,j,k}$ denote the largest value in the domain of $\sigma_{i,j,k}$.

Phase 0. Initialize. Set $x_{0,1,1} = 0$, $\sigma_{i,j,k} = \varnothing$, and $\sigma_{0,1,1}(x_{0,1,1}) = 2\alpha_{0,1,1} + 1$. Extend $\sigma_{0,1,1}$ with more and more values of 0's until $M(\sigma_{0,1,1})$ defined. If $M(\sigma_{0,1,1})$ never becomes defined, then $\sigma_{0,1,1}$ becomes defined on all arguments and is the desired f. Let $q = M(\sigma_{0,1,1})$. Throughout this proof q will denote the current guess of M on the portion f defined so far. We are mostly interested in initial segments on which M changes its mind. We use variables $k_1, k_2, ..., k_m$ to keep track of on which initial segments M changes its mind. Since the first mind change was on $\sigma_{0,1,1}$, we set $k_1 = 1$ (note that M has output one program).

The following algorithm executed in stages will yield the desired f. Each stage of divided into four phases. If some phase fails to terminate we would have the desired f.

For $i = 0$ *to* b *do Begin step* i

Phase 1. Constructing m identical initial segments such that M on that segment outputs m programs.

For $j = 2$ *to* m *do Begin sub step* i, j

1. Set $\sigma_{i,j,k} = \sigma_{i,j-1,1}$ for $k = 1$ to j (initialize $\sigma_{i,j,k}$)
   $\sigma_{i,j,k}(x_{i,j,k} + 1) = 2\alpha_{i,j,k} + 1$ for $k = 1$ to j (place special integers)
2. Now simultaneously execute steps (i) and (ii) below until condition (II) is satisfied.
   (i) For $k \in \{1, ..., j\}$ simultaneously extend $\sigma_{i,j,k}$ with more and more values of $2k-2$ (each $\sigma_{i,j,k}$ start with the same initial segment and branch out differently)
   (ii) Look for the first found $k \in \{1, ..., j\}$ such that $M(\sigma_{i,j,k}) \neq q$, where $q = M(\sigma_{i,j-1,k_{j-1}})$.
3. Set $\sigma_{i,l,k_l} = \sigma_{i,j,k_j}$ for $l = 1, ..., j-1$ (update all the previous segments on which M changed its mind. Now we have j identical segments on which M has changed its mind j times).

Condition (II). If there exists $k \in \{1, ..., j\}$ such that $M(\sigma_{i,j,k}) \neq q$. Let k_j be at least such a k and set $q = M(\sigma_{i,j,k_j})$.

For any i, for each j, M changes its mind on σ_{i,j,k_j}. Otherwise there exists an i and a j for which condition (II) is not satisfied. Then by clause (i) in step (2) the functions $f_{i,j,1}, ..., f_{i,j,j}$ are total. Also by clause (ii) in step (2) $M(f_{i,j,k})\downarrow = q$ for $1 \leqslant k \leqslant j$. By the construction of the above functions they each belong to S. Since $j \geqslant 2$ it cannot be the case that $\varphi_q = {}^c f_{i,j,1}$ and $\varphi_q = {}^c f_{i,j,2}$. Hence the desired f would be either $f_{i,j,1}$ or $f_{i,j,2}$.

End sub step i, j.

Note that for each i, if and when the above loop terminates, $k_1, ..., k_m$ are now defined. $\sigma_{i,1,k_1}, \sigma_{i,2,k_2}, ..., \sigma_{i,m,k_m}$ all have the same initial segment on which M has output m programs. Also each initial segment contains exactly m special integers.

Phase 2. Use the m identical initial segments, diagonalization techniques, and anomaly markers to obtain mind changes.

We will obtain $\lfloor a(m-1)/(c+1-a) \rfloor$ mind changes for M using techniques similar to the ones used in Theorem 4. Let $q = M(\sigma_{i,m,k_m})$. First consider the case when $c+1-a \leqslant a$. We will diagonalize against the current guess of M, using the segments $\sigma_{i,1,k_i}, ..., \sigma_{i,m-1,k_{m-1}}$. Also, we make sure the number of points at which σ_{i,j,k_j} (for $j<m$) is different from σ_{i,m,k_m} is not greater than a. We use the variable *Diag* to keep track of the number of errors on each segment. The variable t is used to keep track of the number of differences created between φ_q and σ_{i,m,k_m}. When t reaches the value $c+1-a$, we will try to force M to change its mind. Set $t=0$.

For $j=1$ *to* $m-1$ *begin sub step* i, j

($\mathscr{A}$) Set $Diag = 0$

($\mathscr{B}$) **While** $Diag < a$ **do**

Execute steps (a) and (b) simultaneously until a y_t is found in (b), and then the rest in order.

(a) Extend σ_{i,j,k_j} with 0's.

(b) Dovetail φ_q on all the points in the extension made to σ_{i,j,k_j} until a convergent point y_t is found.

If program q does not halt, then it is computing a finite function. The function f_{i,j,k_j} is a total function. Let $f = \sigma_{i,m,k_m} \cup \{(x, f_{i,j,k_j}(x)) \mid x > x_{i,m,k_m}\}$. Then $M(f)\downarrow = q$. Therefore $f \notin EX_d^c(M)$. By construction of σ_{i,j,k_j}, $f \in S$, hence we have the desired f. Suppose in step (b) y_t is found.

(c) Set $Diag = Diag + 1$ *and* $t = t+1$

(d) **If** $t = c+1-a$ **then do**

(1) Set

$$\sigma_{i,m,k_m}(x) = \begin{cases} 2 \dot{-} 2\varphi_q(x), & \text{if } x \in \{y_0, ..., y_{t-1}\}; \\ \sigma_{i,j,k_j}(x), & \text{if } x_{i,m,k_m}+1 \leqslant x \leqslant x_{i,j,k_j} \text{ and } x \notin \{y_0, ..., y_{t-1}\}. \end{cases}$$

(2) Set $t=0$

Place a anomaly markers $t_1, ..., t_a$ on consecutive values, starting from the least point not yet in the domain of σ_{i,m,k_m}. Since σ_{i,m,k_m} is defined $\forall x \leqslant x_{i,m,k_m}$, the anomaly markers will be placed at points $x_{i,m,k_m}+1, ..., x_{i,m,k_m}+a$. Execute the steps 3, 4, and 5 simultaneously until condition (IV) or (V) is satisfied.

(3) Extend σ_{i,m,k_m} with more and more values of 0's except at the points where the anomaly markers are kept.

(4) Run program q on points $t_1, ..., t_a$.

(5) Let $\tau = \sigma_{i,m,k_m} \cup \{(x,0) \mid t_1 \leqslant x \leqslant t_a\}$. See if $M(\tau) \neq q$.

Condition (IV). If $\varphi_q(x)\downarrow$ for all such that $t_1 \leqslant x \leqslant t_a$. Then set $\sigma_{i,m,k_m}(x) = 2 \dot{-} 2\varphi_q(x)$ for $t_1 \leqslant x \leqslant t_a$. Extend σ_{i,m,k_m} with more and more values of 0's until $M(\sigma_{i,m,k_m}) \neq q$. If M does not change its mind, on some extension of σ_{i,m,k_m} then f_{i,m,k_m} is a total function and

$M(f_{i,m,k_m})\downarrow = q$. Clearly $f_{i,m,k_m} \in S$, and f_{i,m,k_m} differs at least at $c+1$ points from φ_q. Hence $f_{i,m,k_m} \notin EX_d^c(M)$. Hence the desired f is f_{i,m,k_m}. Suppose M changes its mind on some extension of σ_{i,m,k_m} then set $q = M(\sigma_{i,m,k_m})$. Extend $\sigma_{i,m,k_m}, ..., \sigma_{i,m-1,k_{m-1}}$ to match σ_{i,m,k_m}.

Condition (V). If $M(\tau) \neq q$ then set $q = M(\tau)$ and set $\sigma_{i,m,k_m}(x) = 0$ for $t_1 \leqslant x \leqslant t_a$. Extend $\sigma_{i,1,k_1}, ..., \sigma_{i,m-1,k_{m-1}}$ to match σ_{i,m,k_m}.

Note that if condition (IV) is satisfied then, $\sigma_{i,1,k_1}$ differs from σ_{i,m,k_m} at most at $c+1-a$ points.

Suppose conditions (IV) and (V) are never satisfied, then f_{i,m,k_m} is defined everywhere except at the points $t_1, ..., t_a$. Since condition (IV) was not satisfied there exists some anomaly markers $\{t_{i_1}, ..., t_{i_r}\} \subseteq \{t_1, ..., t_a\}$ for $r \leqslant a$ for which program q does not halt. Hence φ_q is undefined at points $t_{i_1}, ..., t_{i_r}$. Let f be the function defined as follows:

$$f(x) = \begin{cases} f_{i,m,k_m}(x), & \text{if } x \notin \{t_1, ..., t_a\}; \\ 2 \dot{-} 2\varphi_q(x), & \text{if } x \notin \{t_{i_1}, ..., t_{i_r}\}; \\ 0, & \text{otherwise.} \end{cases}$$

Now $M(f)\downarrow = q$ and $f = {}^a f_{i,m,k_m}$. Hence $f \in S$, but $\text{card}(\{x \mid \varphi_q(x) \neq f(x)\}) > c$, therefore $f \notin EX_d^c(M)$. If condition (IV) or (V) is satisfied, we can force M to change its mind again.

End if

End while

(6) Set

$$\sigma_{i,j+1,k_{j+1}}(x) = \begin{cases} 2 \dot{-} \varphi_q(x), & \text{if } x \in \{y_0, ..., y_{t-1}\}; \\ \sigma_{i,j,k_j}(x), & \text{if } x_{i,j+1,k_{j+1}} < x \leqslant x_{i,j,k_j} \wedge x \notin \{y_0, ..., y_{t-1}\} \end{cases}$$

End sub step i, j

If Phase 2 is not completed then we would have the desired f. If phase 2 is completed then M has changed its mind $\lfloor a(m-1)/(c+1-a) \rfloor$ times. If $c+1-a > a$ then the errors are distributed to more than one initial segment. Hence each σ_{i,j,k_j} for $j = 1, ..., m-1$ varies from σ_{i,m,k_m} at most at a points at the end of Phase 2. Let $q = M(\sigma_{i,m,k_m})$.

Phase 3. Using the initial segments force M to change its mind $m-1$ times by diagonalization.

For $j = 1$ *to* $m-1$ *do begin sub step i, j*

1. Set $x = x_{i,j,k_j}$.
2. Simultaneously execute step (a), (b), and (c) for $s = 1, 2, ...$ until condition (B) or (C) is satisfied.
   (a) Set $\sigma_{i,j,k_j}(x+s) = 0$.
   (b) Look for $c+1$ distinct points in the set $\{x+1, ..., x+s\}$ on which φ_q is convergent.
   (c) See if $M(\sigma_{i,m,k_m} \cup (\sigma_{i,j,k_j} - \sigma_{i,m,k_m})) \neq q$.

Condition (B). If there exists $y_0, ..., y_c$ such that $\forall y \in \{y_0, ..., y_c\}\ \varphi_q(y)\downarrow$. Then set

$$\sigma_{i,m,k_m}(z) = \begin{cases} 2 \dot{-} 2\varphi_q(z), & \text{if } z \in \{y_0, ..., y_c\}; \\ \sigma_{i,j,k_j}(z), & \text{if } x_{i,j+1,k_{j+1}} \leqslant z \leqslant x_{i,j,k_j} \wedge z \notin \{y_0, ..., y_c\}. \end{cases}$$

Extend σ_{i,m,k_m} with more and more values of 0's until $M(\sigma_{i,m,k_m}) \neq q$. If such a extension is not found then f_{i,m,k_m} is a total recursive function and $M(f_{i,m,k_m})\downarrow = q$. By the construction of σ_{i,m,k_m} above, $f_{i,m,k_m} \in S$ and $\text{card}(\{x \mid \varphi_q \neq f_{i,m,k_m}\}) > c$, hence $f_{i,m,k_m} \notin EX_d^c(M)$. Therefore the desired f would be f_{i,m,k_m}. Suppose M changes its mind on some extension of σ_{i,m,k_m}, let $q = M(\sigma_{i,m,k_m})$. For $j+1 \leq l \leq m-1$ set

$$\sigma_{i,l,k_l}(z) = \sigma_{i,m,k_m}(z) \qquad \text{for} \quad x_{i,l,k_l} \leq z \leq x_{i,m,k_m}.$$

Condition (C). If $M(\sigma_{i,m,k_m} \cup (\sigma_{i,j,k_j} - \sigma_{i,m,k_m})) \neq q$. Then set $q = M(\sigma_{i,m,k_m} \cup (\sigma_{i,j,k_j} - \sigma_{i,m,k_m}))$. For $j+1 \leq l \leq m$ set

$$\sigma_{i,l,k_l}(z) = \sigma_{i,j,k_j}(z) \qquad \text{for} \quad x_{i,l,k_l} \leq z \leq x_{i,j,k_j}$$

For each j the above loop terminates. Otherwise there exists an i and a j for which conditions (B) and (C) are not satisfied. By clause (a) in step (2) f_{i,j,k_j} is a total function. Let $f = \sigma_{i,m,k_m} \cup (f_{i,j,k_j} - \sigma_{i,m,k_m})$. By clause (c) in step (2) $M(f)\downarrow = q$, and by clause (b) program q computes a finite function. Hence $f \notin EX_d^c(M)$, by construction of σ_{i,j,k_j} $f \in S$; therefore we have the desired f. Hence for each j, M is forced to change its mind.
End sub step i, j

Using m initial segments and diagonalization and anomaly markers we have forced M to change its mind $2m-1+\lfloor a(m-1)/(c+1-a) \rfloor$ times on the segment σ_{i,m,k_m}. We will extend segments identical to σ_{i,m,k_m} so as to diagonalize against M's current guess.

Phase 4. If $i = b$ then skip this phase. Else use the last segment σ_{i,m,k_m} to force M to change its mind $2+\lfloor a/(c+1-a) \rfloor$ times.

Do the following seven steps in order:

(1) Set $\sigma_{i,0,0} = \sigma_{i,m,k_m}$ and $\sigma_{i,0,1} = \sigma_{i,m,k_m}$.
(2) Set $\sigma_{i,0,0} = 2\alpha_{i,0,0}+1$ and $\sigma_{i,0,1} = 2\alpha_{i,0,1}+1$.
(3) Simultaneously extend $\sigma_{i,0,0}$ with more and more values of 0 and $\sigma_{i,0,1}$ with more and more values of 2 until either $M(\sigma_{i,0,0}) \neq q$ or $M(\sigma_{i,0,1}) \neq q$. Suppose M does not change its mind, then $f_{i,0,0}$ and $f_{i,0,1}$ are total recursive functions and $M(f_{i,0,0}) = M(f_{i,0,1}) = q$. But it cannot be the case that $q = f_{i,0,0}$ and $\varphi_q = f_{i,0,1}$. Here the desired f would be either $f_{i,0,0}$ or $f_{i,0,1}$. Hence there should be an extension of $\sigma_{i,0,0}$ or $\sigma_{i,0,1}$ on which M will change its mind. Assume M changed its mind on $\sigma_{i,0,i_0}$, where $i_0 \in \{0, 1\}$ (M is forced to change its mind).
(4) Set $\sigma_{i,m,k_m} = \sigma_{i,0,i_0}$.
(5) If $c+1-a \leq a$ then using the a anomaly markers on $\sigma_{i,0,i_0}$ and the method described in Phase 2, we can force $\lfloor a/(c+1-a) \rfloor$ mind changes for M. Set q to be the last program output by M on $\sigma_{i,0,i_0}$.
(6) Set $x = x_{i,m,k_m}$. Simultaneously execute steps (a), (b), and (c) for $s = 1, 2, \ldots$ until condition (B) or (C) is satisfied.
   (a) Set $\sigma_{i,m,k_m}(x+s) = 0$.
   (b) Look for $c+1-a$ distinct points in the set $\{x+1, \ldots, x+s\}$ on which φ_q is convergent.
   (c) See if $M(\sigma_{i,m,k_m}) \neq q$.

Condition (B). There exists a $y_1, \ldots y_{c+1-a}$ such that $\forall y \in \{y_1, \ldots, y_{c+1-a}\}$ $\varphi_q(y)\downarrow$. Then set

$$\sigma_{i,0,i_0}(x) = \begin{cases} 2 \dot{-} 2\varphi_q(x), & \text{if } x \in \{y_1, \ldots, y_{c+1-a}\}; \\ \sigma_{i,m,k_m}(x), & \text{for } x_{i,0,i_0} \leq x \leq x_{i,m,k_m} \wedge x \notin \{y_1, \ldots, y_{c+1-a}\}. \end{cases}$$

Extend $\sigma_{i,0,i_0}$ with more and more values of 0's until $M(\sigma_{i,0,i_0}) \neq q$. If such a extension is not found then $f_{i,0,i_0}$ is a total function and $M(f_{i,0,i_0})\downarrow = q$. By the construction of $\sigma_{i,0,i_0}$ above, $\text{card}(\{x \mid \varphi_q(x) \neq f_{i,0,i_0}(x)\}) > c$ and the desired f would be $f_{i,0,i_0}$. Hence M must change its mind for some extension of $\sigma_{i,0,i_0}$. Let $q = M(\sigma_{i,0,i_0})$ assuming M has changed its mind.

Condition (C). Suppose $M(\sigma_{i,m,k_m}) \neq q$ then, set $\sigma_{i,0,i_0} = \sigma_{i,m,k_m}$ and $q = M(\sigma_{i,m,k_m})$.
(Note that we have forced M to change its mind.)

(7) Set $\sigma_{i+1,1,1} = \sigma_{i,0,i_0}$.
(We have forced mind changes for M in steps (3), (5), and (6).)

End step i

Clearly, if the above algorithm does not terminate, by the construction there will be an f such that $f \in S$ and $f \notin EX_d^c(M)$. Now if the above algorithm terminates, then M on σ_{b,m,k_m} will output $[\lfloor a(m-1)/(c-a+1) \rfloor (b+1) + 2mb + \lfloor a/(c-a+1) \rfloor b + 2m - 1]$ programs. Let $f = \sigma_{b,m,k_m} \cup \{(x,0) \mid x > x_{a,m,k_m}\}$. Clearly $f \in S$ and $f \notin EX_b^c(M)$ for every $d < [\lfloor a(m-1)/(c-a+1) \rfloor (b+1) + 2mb + \lfloor a/(c-a+1) \rfloor b + 2(m-1)]$. ∎

V. Simulating a Team with Zero Mind Changes and Anomalies with Another Team

LEMMA 8. $(\forall n > 1)(\forall a \in \mathbb{N})[a \geq n-2 \Rightarrow C(n, EX_0^0) \subseteq C(2, EX_a^0)]$.

Proof. Let $n > 1$, $M_1, \ldots, M_n$ be IIMs and $a \in \mathbb{N}$. We will construct two IIMs M_1' and M_2' which will simulate the team of n machines. Suppose n is even. Then $n = 2k$ for some $k \in \mathbb{N}$. Divide the team of n machines into two subteams each consisting of k machines. Now by Lemma 3 if $a \geq 2k-2$ then $C(k, EX_0^0) \subseteq C(1, EX_a^0)$. Make M_1' and M_2' simulate each subteam independently, hence if $a \geq 2k-2$ then $C(n, EX_0^0) \subseteq C(2, EX_a^0)$. Since $n = 2k$ the result follows. Suppose n is odd. Then $n = 2k+1$ for some $k \in \mathbb{N}$. By Lemma 3, for M_1' and M_2' to have the maximum number of mind changes the team of n members must output n programs, say $p_1, \ldots, p_k, \ldots, p_{2k+1}$. M_1' and M_2' simulate the team of n members using the following algorithm. M_1' outputs $p_1, p_{k+2}, \{p_{k+2}, p_{k+3}\}, \ldots, \{p_{k+2}, \ldots, p_{2k+1}\}$. M_1' has output $1+k$ programs so far. Then M_1' has output $1+k$ programs so far. Then M' proceeds to eliminate $k-1$ programs from the amalgamation $\{p_{k+2}, \ldots, p_{2k+2}\}$. Hence M_1' beeds to output at most a total of $2k$ programs. That is it needs $2k-1$ mind changes, since $n = 2k+1$ M_1' needs $n-2$ mind changes. M_2' outputs $\{p_1, p_2\}, \{p_1, p_2, p_3\}, \ldots, \{p_1, \ldots, p_{k+1}\}$. So far M_2' has output k programs. But the amalgamation $\{p_1, \ldots, p_{k+1}\}$ has $k+1$ programs. Hence it needs to output at most another k programs to eliminate the wrong programs from the amalgamation. Hence in all it, needs $2k-1 = n-2$ mind changes. Hence M_1' and M_2' together can simulate a team of n members with $n-2$ mind changes. ∎

THEOREM 9. $(\forall m > n > 0)(\forall a \in \mathbb{N})[C(m, EX_0^0) \subseteq C(n, EX_a^0)$ *if and only if* $a \geq \lceil 2m/n \rceil - 2]$.

Proof. ($\Leftarrow$) Let $a \in \mathbb{N}$, $m > n > 0$, and $M_1, M_2, ..., M_m$ be a team of IIMs. Call this team the M team. We will construct IIMs $M'_1, ..., M'_n$ (the M' team) which will simulate the team of m machines.

Case 1. n divides m ($m \bmod n = 0$). Divide the m team members into n equal subteams. Let each member of the M' team simulate one of the subgroups, using the algorithm given in Lemma 3. Since each subteam has m/n IIMs, each member of the M' team needs $2m/n - 2 = \lceil 2m/n \rceil - 2$ mind changes.

Case 2. n does not divide m. Suppose n is even. Then $n = 2k$ for some $k \in \mathbb{N}$. Divide the m member team into k subteams with each subteam consisting of at most $\lceil m/k \rceil$ IIMs. By Lemma 8, each subteam can be simulated using two IIMs from the M' team with $\lceil m/k \rceil - 2$ mind changes. But $\lceil m/k \rceil = \lceil 2m/2k \rceil = \lceil 2m/n \rceil$. Hence, each member of the M' team requires $\lceil 2m/n \rceil - 2$ mind changes. Suppose n is odd. Then $n = 2k + 1$ for some $k \in \mathbb{N}$. Divide the m member team into k subteams, with each subteam having $\lceil 2m/n \rceil$ IIMs. There are $m - k\lceil 2m/n \rceil$ IIMs from the m member team which do not belong to any subteam. Usetwo IIMs from the n member team to simulate each subteam and one IIM from the n member team to simulate $m - k\lceil 2m/n \rceil$ left over IIMs from the m member team. By Lemma 8, $\lceil 2m/n \rceil - 2$ mind changes are needed to simulate the subteams. By Lemma 3, $2(m - k\lceil 2m/n \rceil) - 2$ mind changes are needed to simulate $m - k\lceil 2m/n \rceil$ IIMs. $2m - 2k\lceil 2m/n \rceil = 2m - (n-1)\lceil 2m/n \rceil = 2m - n\lceil 2m/n \rceil + \lceil 2m/n \rceil$. But $2m - n\lceil 2m/n \rceil \leq 0$, hence $2m - 2k\lceil 2m/n \rceil \leq \lceil 2m/n \rceil$. Consequently, the M' team can simulate the M team using $\lceil 2m/n \rceil - 2$ mind changes.

($\Rightarrow$) Let $m > n > 0$ and $a \in \mathbb{N}$. Let $M_1, ..., M_n$ be IIMs. We will prove the contrapositive: if $a < \lceil 2m/n \rceil - 2$ then $C(m, EX_0^0) - C(n, EX_a^0) \neq \varnothing$. The following is an intuitive description of S, a set of recursive functions. Given a $f \in S$ there will be $m - 1$ other recursive functions in S which will have the same initial segment as f. In that common initial segment there will be at most m special integers one of which will describe f. We will choose the special integers to be odd integers. Formally, $f \in S$ if and only if the following two conditions hold.

1. $f(x)$ is odd for no more than m distinct values of x.
2. There exists x and j such that $f(x) = 2j + 1$ and $\varphi_j = f$.

We will construct IIMs $M'_1, ..., M'_m$ such that $S \subseteq EX_0^0(M'_1, ..., M'_m)$. For any $f \in S$, ith member of the above team on input f executes the following algorithm.

Begin M_i'
1. Wait for the ith odd integer, say k.
2. Output $(k-1)/2$.

End M_i'

By the definition of S, for any $f \in S$ one of the m special integers in f will describe f. Therefore $S \in C(m, EX_0^0)$. We will construct an f such that, $f \in S$ and $f \notin EX_a^0(M_1, ..., M_n)$. Without loss of generality we can assume that, each M_i (for $1 \leqslant i \leqslant n$) outputs at most $a+1$ guesses. From now on the word team will refer to the IIMs, $M_1, ..., M_n$. Given a program $\alpha_{j,k}$, we will abbreviate $\varphi_{\alpha_{j,k}}$ by $f_{j,k}$. Using a $m(m+1)/2$-ary recursion theorem and a finite extension argument we construct functions $f_{j,k}\, j \in \{1, ..., m\}$ and $k \in \{1, ..., j\}$. For $j \in \{1, ..., m\}$, and $k \in \{1, ..., j\}$ let $\sigma_{j,k}$ denote the largest initial segment of $f_{j,k}$ constructed so far. Let $x_{j,k}$ denote the largest value in the domain of $\sigma_{j,k}$.

Phase 0. Initialize. Set $x_{1,1} = 0$, $\sigma_{j,k} = \varnothing$, and $\sigma_{1,1}(x_{1,1}) = 2\alpha_{1,1} + 1$. We will use variables $i_1, ..., i_n$ to keep track of the last program output by the IIMs $M_{i_1}, ..., M_{i_n}$, respectively. Extend $\sigma_{1,1}$ with more and more values of 0's until $M_i(\sigma_{1,1})$ defined for some $1 \leqslant i \leqslant n$. Let i_1 be that i. Let $q_{i_1} = M_{i_1}(\sigma_{1,1})$. Throughout this proof q_i will denote the current guess of M_i (for $1 \leqslant i \leqslant n$) on the portion f defined so far. We are mostly interested in inital segments on which the team changes its mind. We use variables $k_1, k_2, ..., k_m$ to keep track of on which initial segments the team changes its mind. Since the first mind change was on $\sigma_{1,1}$, we set $k_1 = 1$. Note that team has output one program.

The following algorithm executed in stages will yield the desired f. Each phase is divided into stages. If some stage fails to terminate we would have the desired f.

Phase 1. Construct m identical initial segments such that the M team on that segment outputs m programs. The Phase 1 construction is similar to the Phase 1 construction of Theorem 4.

For $j = 2$ *to* m *do Begin stage* j
1. Set $\sigma_{j,k} = \sigma_{j-1,1}$ for $k = 1$ to j (initialize $\sigma_{j,k}$)
   Define $\sigma_{j,k}(x_{j,k} + 1) = 2\alpha_{j,k} + 1$ for $k = 1$ to j (place special integers)
2. Simultaneously execute step (i) and (ii) below until condition (II) is satisfied.
   (i) For $k \in \{1, ..., j\}$ simultaneously extend $\sigma_{j,k}$ with more and more values of $2k-2$ (each $\sigma_{j,k}$ starts with the same initial segment and branch out differently)
   (ii) Look for the least $k \in \{1, ..., j\}$ such that $M_i(\sigma_{j,k}) \notin \{q_1, ..., q_n\}$ for some $1 \leqslant i \leqslant n$, where $q_1, ..., q_n$ are the current guess by each team member.
3. Set $\sigma_{l,k_l} = \sigma_{j,k_j}$ for $l = 1, ..., j-1$ (update all the previous segments on which the M team changed its mind.)

Condition (II). There exists $k \in \{1, ..., j\}$ and an $1 \leqslant i \leqslant n$ such that $M_i(\sigma_{j,k}) \notin \{q_1, ..., q_n\}$. Let k_j be least such k and i_j be the corresponding i, set $q_{i_j} = M_{i_j}(\sigma_{j,k_j})$.

For each j, one member of the team chages its mind on σ_{j,k_j}. Otherwise there exists a j for which condition (II) is not satisfied. Then by clause (i) in step (2) the functions $f_{j,1}, ..., f_{j,j}$ are total and distinct. Also by clause (ii) in step (2) for each $i \leqslant n$, for all $k \leqslant j$, $M_i(f_{j,k})$ has converged to program q_i. If $j > n$ then the team has only n guesses, since there are j distinct functions, there exists a j_0 such that the team cannot identify f_{j,j_0}. If $j \leqslant n$ then the team had output only $j-1$ guesses. Since there are j distinct functions there exists a j_1 such that f_{j,j_1} is not identified by the team. By the construction each of the above functions belongs to S. Hence the desired f would be either f_{j,j_0} or f_{j,j_1}.
End step j.

Note that if and when the above loop terminates, $k_1, ..., k_m$ are now defined. $\sigma_{1,k_1}, \sigma_{2,k_2}, ..., \sigma_{m,k_m}$ all have the same initial segment on which team has output m programs. Also, each initial segment contains exactly m special integers.

Phase 2. Using the initial segments, force the team to output $m-n$ programs by diagonalization. We have to diagonalize against n programs, variable *count* is used to keep track of how many programs we have diagonalized against so far. Let $D = \{q_1, ..., q_n\}$, $count = 0$.

For $j = 1$ *to* $m-1$ *do begin stage j*

1. Set $x = x_{j,k_j}$.
2. Simultaneously execute step (a), (b), and (c) for $s = 1, 2, ...$ until condition (B) or (C) is satisfied.
   (a) Set $\sigma_{j,k_j}(x+s) = 0$.
   (b) Look for a y in the set $\{x+1, ..., x+s\}$ on which $\varphi_q(y)$ is convergent where $q \in D$.
   (c) See if there exists an i such that $M_i(\sigma_{j,k_j}) \notin \{q_1, ..., q_n\}$.

Condition (B). There exists $q \in D$ and a point y such that $\varphi_q(y)\downarrow$. Then set $D = D - \{q\}$, $count = count + 1$ and

$$\sigma_{j+1,k_{j+1}}(z) = \begin{cases} 2 \dot{-} 2\varphi_q(z), & \text{if } z = y; \\ \sigma_{j,k_j}(z), & \text{if } x_{j+1,k_{j+1}} \leqslant z \leqslant x_{j,k_j} \wedge z \neq y. \end{cases}$$

IF $count = n$ **then**

Extend $\sigma_{j+1,k_{j+1}}$ with more and more values of 0's until $M_i(\sigma_{j+1,k_{j+1}}) \notin D$ for some $i \leqslant n$. If such a extension is not found then $f_{j+1,k_{j+1}}$ is a total recursive function and $M_i(f_{j+1,k_{j+1}})\downarrow = q_i$ for all $i \leqslant n$. Also, by construction $\varphi_{q_i} \neq f_{j+1,k_{j+1}}$, for all $1 \leqslant i \leqslant n$. Hence $f_{j+1,k_{j+1}} \notin C(n, EX_a^0)$. By the construction of $\sigma_{j+1,k_{j+1}}$ above, $f_{j+1,k_{j+1}} \in S$. Therefore, the desired f would be $f_{j+1,k_{j+1}}$. Suppose there exists some i such that M_i changes its mind on some extension of $\sigma_{j+1,k_{j+1}}$, set $q_i = M_i(\sigma_{j+1,k_{j+1}})$, $count = count - 1$ and $D = D \cup \{q_i\}$.

End If

Condition (C). If there exists a $i \leqslant n$ such that $M_i(\sigma_{j,k_j}) \notin D$. Then set $D = D - \{q\}$, $q_i = M_i(\sigma_{j,k_j})$, $count = count - 1$, $D = D \cup \{q_i\}$ and

$$\sigma_{j+1,k_{j+1}}(z) = \sigma_{j,k_j}(z) \qquad \text{for} \quad x_{j+1,k_{j+1}} \leqslant z \leqslant x_{j,k_j}$$

End step j

Suppose the above for loop does not terminate, then there exists a j for which Conditions (B) and (C) is not satisfied. By clause (a) in step (2) f_{j,k_j} is a total recursive function. Let $f = f_{j,k_j}$. By clause (c) in step (2) $M_i(f)$ converges to q_i, and by clause (b) each q_i computes a finite function. Hence $f \notin C(n, EX_a^0)$, by construction of $\sigma_{j,k_j} f \in S$, therefore we have the desired f. Suppose the above loop terminates, then the team outputs $m-n$ programs. Now on Phase 1, the team guessed m programs, hence the team has guessed a total of $2m-n$ programs. Now there a n members in the team, hence there is a member of the team which has guessed $\lceil (2m-n)/n \rceil$ programs, i.e., $\lceil 2m/n \rceil - 1$ programs. Hence if $a < \lceil 2m/n \rceil - 2$ then $C(n, EX_0^0) - C(2, EX_a^0) \neq \varnothing$. ∎

VI. Conclusions

Simulation of a team of m IIMs with a single IIM was studied. Theorem 2 indicates that a single IIM with less than $2(m-1)$ mind changes can never simulate a team of IIMs. Theorem 4 shows the exact trade-off between mind changes and anomalies for the single IIM when the team members are allowed zero mind changes. Theorem 7 generalizes all the above gives a precise bound on the number of mind changes and the number of anomalies for one IIM to simulate a team of IIMs.

We have not completely succeeded in our goal of characterizing the trade-offs between a bounded number of anomalies, a bounded number of mind changes and a fixed number of inference machines. For example, our results tell us that $C(3, EX_0^2) \subseteq C(2, EX_3^2)$. This is seen by breaking the three member team up into a two member team which can be simulated by a EX_3^2 type inference machine, and a single inference machine was, by definition can be simulated by the other EX_3^2 type inference machine. However, our results do not tell us if $(3, EX_0^2) - C(2, EX_2^2) \neq \varnothing$?

Acknowledgments

We thank the referees for suggesting several improvements in the exposition. Our colleague W. I. Gasarch made several comments on various drafts of this paper.

Received May 7, 1987; accepted June 30, 1988

References

Angluin, D., and Smith, C. H. (1983), Inductive inference: Theory and methods, *Comput. Surveys* **15**, 237–269.

ANGLUIN, D., AND SMITH, C. H. (1987), Inductive inference, *in* "Encyclopedia of Artificial Intelligence" (S. Shapiro, Ed.), pp. 409–418, Wiley, New York.

BARZDIN, J. (1974), Two theorems on the limiting synthesis of functions, *in* "Theory of Algorithms and Programs" (Barzdin, Ed.), Vol. 1, 82–88, Latvian State University, Riga, USSR.

BARZDIN, J. A., AND FREIVALDS, R. V. (1972), On the prediction of general recursive functions, *Soviet Math. Dokl.* **13**, 1224–1228.

BLUM, L., AND BLUM, M. (1975), Toward a mathematical theory of inductive inference, *Inform. and Control* **28**, 125–155.

CASE, J. (1974), Periodicity in generations of automata, *Math. Systems Theory* **8**, 15–32.

CASE, J., AND NGO MANGUELLE, S., Refinements of inductive inference by popperian machines, *Kybernetika.*

CASE, J., AND SMITH, C. (1983), Comparison of identification criteria for machine inductive inference, *Theoret. Comput. Sci.* **25**, 193–220.

DALEY, R. P., AND SMITH, C. H. (1986), On the complexity of inductive inference, *Inform. and Control.* **69**, 12–40.

GOLD, E. M. (1967), Language identification in the limit, *Inform. and Control* **10**, 447–474.

JANTKE, K. P. (1979), Natural properties of strategies identifying recursive functions. *Electron. Inform. Kybernet.* **15**, 487–496.

KINBER, E. B. (1977), On a theory of inductive inference, *in* "Lecture Notes in Computer Science, Vol. 56," pp. 435–440, Springer-Verlag, New York/Berlin.

MACHTEY, M., AND YOUNG, P. (1978), "An Introduction to the General Theory of Algorithms," North-Holland, New York.

PITT, L. (1984), A characterization of probabilistic inference, *in* "Proceedings, 25th Annual Symposium on Foundations of Computer Science, Palm Beach, FL," pp. 485–494.

PITT, L., AND SMITH, C. (1988), Probability and plurality for aggregations of learning machines, *Inform. and Comput.* **77**, 77–92.

SMITH, C. H. (1982), The power of pluralism for automatic program synthesis, *J. Assoc. Comput. Mach.* **29**, 1144–1165.

VALIANT, L. G. (1984), A theory of the learnable, *Comm. ACM* **27**, 1134–1142.

VELAUTHAPILLAI, M. (1986), "On the Inductive Inference of Programs with Anomalies," Ph.D. thesis, The University of Maryland, College Park, MD.

WIEHAGEN, R., FREIVALDS, R., AND KINBER, E. K. (1984), On the power of probabilistic strategies in inductive inference, *Theoret. Computer. Sci.* **28**, 111–133.